Amendments to the Claims:

This listing of claims replaces all prior versions, and listings, of claims in this application.

Listing of Claims:

1. (Previously presented) A computerized method for detecting anomalous behavior in an executing software program, said method comprising the steps of :

generating a normal execution trace for the software program;

applying a learning algorithm to the normal execution trace to build a finite automaton;

applying an examination algorithm to the finite automaton to identify undesirable transition states in the finite automaton and to create a labeled finite automation; and

applying the labeled finite automaton to an execution trace associated with the executing software program.

2. (Currently amended) The method of claim 1, wherein application of the learning algorithm occurs during a first learning, phase and application of the examination algorithm occurs during a second, examination phase and the learning phase occurs before and ~~discretely~~ independently from the examination phase.

3. (Previously presented) The method of claim 1, wherein the labeled finite automaton identifies undesirable behavior.

4. (Currently amended) The method of claim 3, wherein application of the labeled finite automaton to the execution trace is performed to identify undesirable behavior in a ~~novel~~ execution trace.

5. (Previously presented) The method of claim 4, wherein the undesirable behavior comprises the undesirable transition.

6. (Previously presented) The method of claim 1, wherein the finite automaton comprises a tuple.

7. (Previously presented) The method of claim 6, wherein the tuple comprises a set of possible states, a set of symbols comprising the input alphabet, a mapping function, a start state, and a set of final states.

8. (Previously presented) The method of claim 6, wherein the tuple comprises a set of states interconnected by labeled transitions.

9. (Previously presented) The method of claim 1, wherein the finite automaton comprises a prefix tree.

10. (Previously presented) The method of claim 9, wherein the prefix tree comprises a plurality of nodes.

11. (Previously presented) The method of claim 10, wherein the plurality of the nodes of the prefix tree correspond to states of the finite automaton.

12. (Previously presented) The method of claim 11, wherein one of the plurality of nodes comprises a root node, with the root node serving as a start state.

13. (Previously presented) The method of claim 12, wherein a remainder of the plurality of nodes comprise accepting states.

14. (Previously presented) The method of claim 10, wherein the learning algorithm selectively merges nodes in the finite automaton.

15. (Previously presented) The method of claim 1, wherein the learning algorithm comprises a state merging algorithm.

16. (Previously presented) The method of claim 1, further comprising flagging the execution trace associated with the executing software program as malicious if the execution trace associated with the executing software program is rejected by the finite automaton.

17. (Previously presented) The method of claim 1, wherein the execution trace associated with the executing software program is rejected if it does not end in an accepting state.

18. (Previously presented) The method of claim 3, wherein the undesirable behavior comprises at least one of providing an undesired method of entry into the system to unauthorized users, damaging system resources, and elevating user privileges.

19. (Previously presented) The method of claim 1, wherein the finite automaton is built using empirical data.

20. (Previously presented) The method of claim 1, wherein the method is performed by monitoring processes at a system level.